relationalAI

Database Repairs and Consistent Query Answering:

Origins and Further Developments

Leo Bertossi

Also: IMFD Chile

ACM PODS 2019

relationalAI

ACM PODS 1999: "Consistent Query Answers in Inconsistent Databases"

Marcelo Arenas, Leo Bertossi and Jan Chomicki

First Questions

(with Marcelo Arenas and Jan Chomicki)

- Given a database schema $\mathcal{S}$, including a set of integrity constraints Σ

  Two instances for $\mathcal{S}$, not necessarily consistent wrt. Σ

  How can be compare them wrt. inconsistency?

- Given instance D for $\mathcal{S}$:

  If $D \not\models \Sigma$, how (much) inconsistent is D wrt. Σ?

  In quantitative terms?

  Can we compare instances on this basis?

- We did not develop these ideas as such, but asked a related question ...

Second Questions

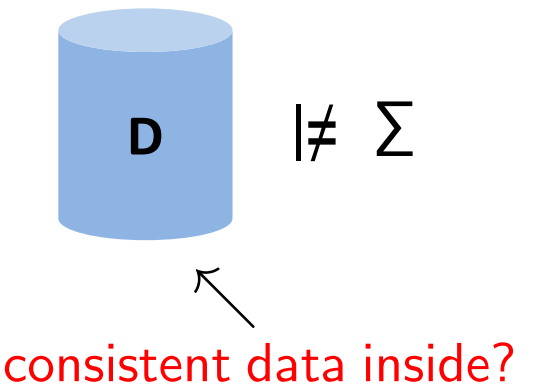
- We may want to compare instances wrt. their "contents of consistent data"

  What are the consistent data in an inconsistent DB?

- Can we characterize those data?

  Can we extract those data?

  Via queries posed to the inconsistent DB at hand?



- Characterization?

  Consistent data are those that are "invariant under all ways of restoring consistency"

  Not quite precise: Need a "repair semantics"

  Repair actions? How far should be go? (as close as possible)

Database Repairs and Consistent Query Answers

Example 1: Denial constraints (DCs) and an inconsistent DB D

$$\neg\exists x\exists y(P(x) \wedge Q(x,y))$$
$$\neg\exists x\exists y(P(x) \wedge R(x,y))$$

P	A
	a
	e

Q	A	B
	a	b

R	A	C
	a	c

Repair semantics: (a) Tuple deletions under set inclusion (b) Set of deleted tuples minimal

P	A
	e

Q	A	B
	a	b

R	A	C
	a	c

D_1

P	A
	a
	e

Q	A	B

R	A	C

D_2

Two subset-repairs (S-repairs) $\subseteq$-maximal consistent sub-instances of D

$P(e)$ invariant

- Repair semantics for general ICs?

  Tuple insertions and deletions

  S-repairs D' make $D \Delta D' := (D \smallsetminus D') \cup (D' \smallsetminus D)$ subset-minimal

- Several others were introduced and investigated later ...

  An interesting one: cardinality-repairs (C-repairs): minimize $|D \Delta D'|$

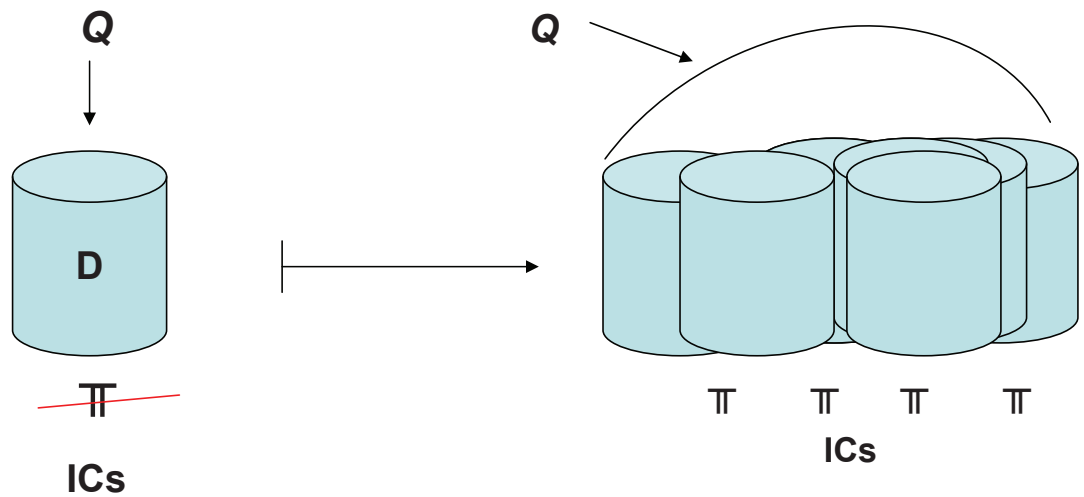
  Previous example: Only D_1 is C-repair

- Not only atomic data can be consistent, but also combinations thereof

  Query answers ...

  What are the consistent answers to a query $\mathcal{Q}$ posed to D?

- Extend the idea: Answers that are invariant under the class of S-repairs



Answers to $\mathcal{Q}$ that are simultaneously returned by all S-repairs

Previous example: Consistent answers to query $\mathcal{Q}(x)$: $P(x) \vee \exists y Q(x,y)$?

From: D_1 : $\{a, e\}$ D_2 : $\{a, e\}$ Consistent answers: $\{a, e\}$

- Computing, materializing, querying all repairs did not look promising ...

Starting to Compute Consistent Answers

- Can we obtain the consistent answers to $\mathcal{Q}$ by querying the inconsistent D?

  Most of initial research was motivated by this question (and characterization of consistent data)

  Not by repair computation; even less of a single repair (data cleaning)

- First idea: Take inspiration from "semantic query optimization" (SQO)

  Previous example, with DCs: $\neg\exists x\exists y(P(x) \wedge Q(x,y))$, $\neg\exists x\exists y(P(x) \wedge R(x,y))$

  Query: $\mathcal{Q}(x)$: $P(x)$?

  If D consistent with DCs, $\mathcal{Q}$ equivalent to $\mathcal{Q}'(x)$: $P(x) \wedge \neg\exists y Q(x,y) \wedge \neg\exists y R(x,y)$

  In SQO residues from DCs as extra conditions may speed up QA

- If D inconsistent, residues impose extra local conditions: pose $\mathcal{Q}'$ to D

What Came Next?

- In the PODS'99 paper we investigated the first-order query rewriting approach: soundness, completeness, termination, ...
- Clearly it had limited applicability: non-termination, infinite rewritings, exponentially many repairs (suggesting high complexity), ...
- Several issues started to be investigated:
  - Intrinsic complexity of consistent query answering (CQA)

    For certain classes of ICs and queries, e.g. FDs and conjunctive queries

    CQA may be coNP-complete
  - Algorithms for CQA
  - More expressive languages for query rewriting

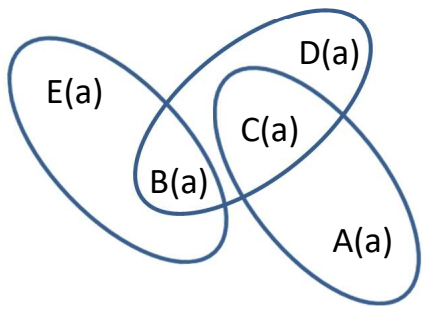
- Useful connection with hyper-graphs, for complexity analysis and algorithms

For FDs, DCs, ..., tuple-deletion based repairs

Example 2: $D = \{A(a), B(a), C(a), D(a), E(a)\}$

$$\Sigma = \{\neg\exists x(B(x) \wedge E(x)),\ \neg\exists x(B(x) \wedge C(x) \wedge D(x)),\ \neg\exists x(A(x) \wedge C(x))\}$$

Conflict hyper-graph (CHG):



S-repairs: $\subseteq$-maximal independent sets:

$D_1 = \{B(a), C(a)\}$, $D_2 = \{C(a), D(a), E(a)\}$, $D_3 = \{A(a), B(a), D(a)\}$

C-repairs: maximum-cardinality independent sets: $D_2,\ D_3$

Obtained removing vertices in minimum-size hitting-sets of hyper-edges

- Hyper-edges bounded in size (depend on DCs) $\rightsquigarrow$ approximations, FPT

ASP-Based Specification of Repairs

- ASPs can be used to specify, compute and query S- and C-repairs (disjunctive logic programs with stable model semantics)
- ASP can be seen as the expressive language for query rewriting

Example 3: DC: $\kappa : \neg\exists x \exists y (S(x) \wedge R(x,y) \wedge S(y))$ DB D

R	A	B
	a_4	a_3
	a_2	a_1
	a_3	a_3

S	A
	a_4
	a_2
	a_3

Repair-ASP contains D as set of facts plus the rules:

$$S'(x,\mathsf{d}) \vee R'(x,y,\mathsf{d}) \vee S'(y,\mathsf{d}) \leftarrow S(x), R(x,y), S(y),$$
$$S'(x,\mathsf{s}) \leftarrow S(x),\ not\ S'(x,\mathsf{d}).$$

etc.

d, s: annotations for "deleted" and "stays (in repair)", resp.

- A stable model M of the program determines an S-repair D' of D:

  $D' \ := \ \{R(\bar{c}) \mid R'(\bar{c},\mathsf{s}) \ \in \ M\}$ (and every S-repair obtained in this way)

  An S-repair D_1 represented by model

  $M_1 = \{R'(a_4, a_3, \mathsf{s}), R'(a_2, a_1, \mathsf{s}), R'(a_3, a_3, \mathsf{s}), S'(a_4, \mathsf{s}), S'(a_2, \mathsf{s}), S'(a_3, \mathsf{d}), \ldots\}$

- For sets of DCs (including FDs) repair programs can be made non-disjunctive

  Maybe non-stratified, e.g. for FDs and DCs with self-joins

- Complexity of ASPs match exactly the intrinsic complexity of repair-related computations and CQA

- CQA becomes QA under ASPs, which can be optimized with magic-set methods

- Models corresponding to C-repairs can be obtained by adding weak program constraints (WCs)

Example: (3 cont.) Add WCs

$$\Leftarrow R(\bar{x}), R'(\bar{x}, \mathsf{d})$$
$$\Leftarrow S(\bar{x}), S'(\bar{x}, \mathsf{d})$$

Keep only models that minimize the number of violations of WCs

Here: minimize the number of deleted tuples

- Repair-ASPs are a uniform formalism for specifications of repairs and CQA
- Can be integrated with others: virtual data integration, peer-data exchange, ontologies, ...
- Shed light on complexity cases via ASP analysis

Some Other Developments

A. Repair Semantics

- A particular repair semantics considered in the Pods'99 paper

  S- and C-repairs extensively investigated

- Other repair semantics have been considered
- Two that have been considered in certain applications

Example 4: Inconsistent DB D wrt. DC $\neg\exists x\exists y(S(x) \wedge R(x,y) \wedge S(y))$

R	A	B
	a_4	a_3
	a_2	a_1
	a_3	a_3

S	A
	a_4
	a_2
	a_3

Repair by "minimally" changing attribute values

To avoid satisfying the join

R	A	B
	a_4	a_3
	a_2	a_1
	a_3	a_3

S	A
	a_4
	a_2
	NULL

R	A	B
	a_4	NULL
	a_2	a_1
	a_3	NULL

S	A
	a_4
	a_2
	a_3

NULL given a semantics as in SQL DBs

Minimize sets of attribute-value changes under set inclusion (or in cardinality)

Repair ASP programs can be produced

Example 5:

$Supply$	Company	Receiver	Item
	C_1	R_1	I_1
	C_2	R_2	I_2
	C_2	R_1	I_3

$Articles$	Item	Cost
	I_1	50
	I_2	30

Inconsistent wrt. inclusion dependency $\forall x \forall y \forall z (Supply(x, y, z) \rightarrow \exists v\ Articles(z, v))$

Insert tuples with NULL

As above ...

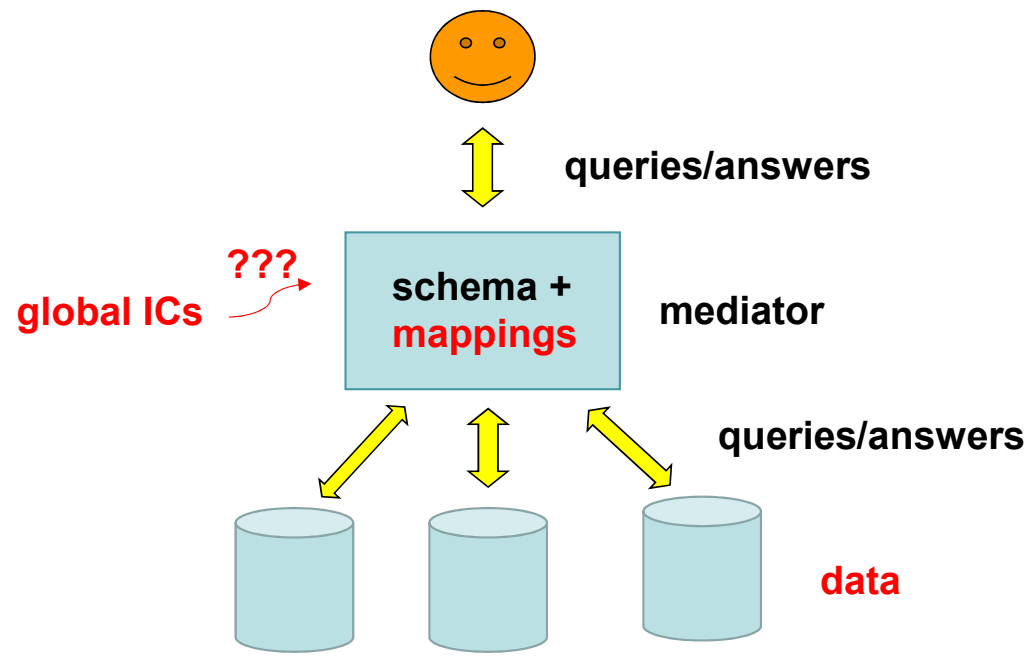
$Articles$	Item	Cost
	I_1	50
	I_2	30
	I_3	NULL

B. Virtual Data Integration

- Independent and autonomous sources, integrated via a mediator
- Mediator offers a schema and a DB-like interface, but no material data
- Mediator receives user queries and collects data from sources via queries
- Possible-world semantics: legal global instances
- Global ICs cannot be enforced on source data, only on query answers

Perfect application scenario for CQA

- Repair-ASPs can be combined with LP-based specifications of legal instances



C. Inconsistency-Tolerant OBDA

- Data sources queried via an ontology

  Through a conceptual model

  Ontology-based data access

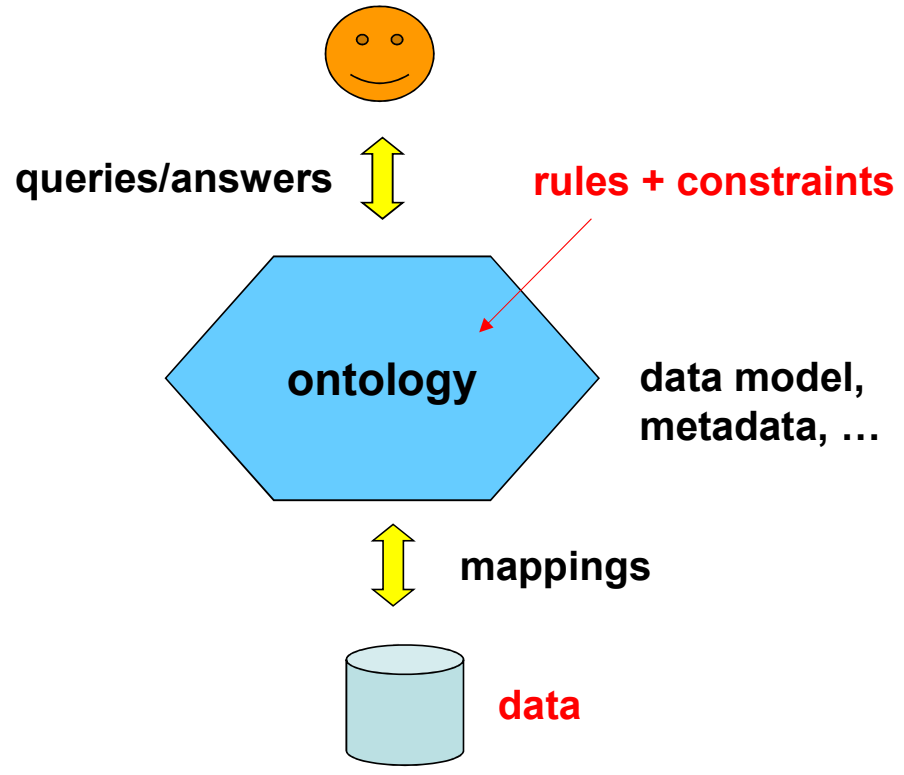
- Ontology contains rules and constraints, e.g. in Description Logic, Datalog$^{\pm}$

- Combination with underlying data may lead to inconsistency

  Repairs are applied

  Usually at the data level

- Newer area where much is happening



The Initial Question: Inconsistency Measures for DBs

- Use repair semantics as a basis for inconsistency measures, e.g.

$$inc\text{-}deg^C(D,\Sigma) \quad := \quad \frac{|D| - max\{|D'| \ : \ D' \in C\text{-}Rep(D,\Sigma)\}}{|D|}$$

- $0 \leq inc\text{-}deg^C(D,\Sigma) \leq 1$, with value 0 when D consistent
- One C-repair is good enough

Example: (1 cont.) $C\text{-}Rep(D,\Sigma) = \{D_1\}$ $\qquad inc\text{-}deg^C(D,\Sigma) = \frac{4-|D_1|}{4} = \frac{1}{4}$

- ASPs with WCs have exactly the required expressive power/complexity needed for IM computation
- IM can be computed via $|D \setminus D'|$ for some (any) C-repair D'

A simple aggregation and a query to ASP [Bertossi, LPNMR'19]

Final Remarks

- Repairs and CQA have been extensively investigated since the PODS'99 paper

  Much beyond the original setting of a stand alone database under classical ICs
- Repairs and CQA not initially conceived as a data cleaning approach or solution

  Almost "the opposite": Use "dirty" DB to extract "clean" data via queries

  Repairs and CQA have had impact on data cleaning, starting by illuminating the need for a precise "cleaning semantics"
- When a DB is cleaned according to quality criteria (maybe not consistency):
  - What are the intended clean instances?
  - If one clean instance is computed, how does it relate to that class?
  - What are the intrinsically clean data in the dirty instance?

- We have shown some areas of application, among others

  The recently unveiled connection between database repairs and causality in DBs has been used to provide complexity and algorithms for the latter

- There are many open problems and (almost) unexplored research directions

  - We need system implementations and applications

    Use approximations for the hard cases of CQA

  - Investigate repair semantics in more general terms

    What is the right repair semantics satisfying certain properties?

  - Dynamic aspects

    Complexity of- and algorithms for problems in the presence of updates

    Incremental repair computation under updates

    Incremental CQA under updates

Thanks for listening!

Thanks for the "Gems of PODS" recognition!

Thanks to my coauthors, Marcelo and Jan!

Thanks to all the colleagues who have worked in the area; it is theirs now ...

My new academic domicile:

Universidad Adolfo Ibáñez (UAI), Santiago, Chile

I couldn't get any higher in my career!

(literally ..., on top of the Andes mountains) https://www.uai.cl